Can a Computer Learn Physics? Learning the Magnetization of the Ising Model with Conditional Variational Autoencoders

Juan Diego Castro Miyashiro

Department of Computing Science

University of Alberta

"That what I cannot create I cannot understand"

-Richard Feynman

Abstract-

Conditional Variational Autoencoders are a type of Generative method that allows you to infer the underlying distribution of an input dataset in addition to provide an easy mechanism to sample from it. In this project we explore if we can use this technique to learn the Boltzmann-Gibbs distribution of the Ising Model. We used the Metropolis-Hasting algorithm to generate Ising Lattice images and to serve as a computational benchmark for the CVAE. The measure of success was the ability to predict the magnetization curve. Our best results were obtained by a fully connected CVAE with a dimensionality of Z equal to 4

Introduction

Can a computer learn the underlying distribution of a Condensed Matter Physics model? In Physics the behavior of materials at thermal equilibrium is modeled by a probability distribution over the possible configurations of the physical system. It is asserted that the macroscopic properties we experience in our daily life are expectation values of this distribution over the space of all the different possible configurations of those systems. This distribution for non-quantum bodies is the Boltzmann-Gibbs distribution. Naively, this means that the only thing we need to do to know how any observable changes over temperature is to device some type of sampling scheme (such as rejection sampling for instance) and take the empirical average of the data to approximate the observable. After all the Boltzmann-Gibbs distribution is a member of the exponential family [1]. In the same way, it also possible, at least in theory, to brute-force compute the expression for the desired expectation given some temperature. In practice however both approaches prove incredibly challenging if not outright impossible when we look at dimensionality. Even for a small model of a solid lattice with 30x30 binary variable the space grows exponentially with 2^900 possible configurations thus the problem becomes intractable for all but the smallest of the instances.

Historically the canonical way to tackle this problem has been using the Metropolis-Hasting algorithm; a MCMC (Markov Chain Monte Carlo) method that explores and samples the state space around the neighborhood of areas with more probability density more frequently than states nearby low-density areas [2]. These methods however have the curse of requiring long burn-in sequences to avoid correlated sampling and having a relatively high accuracy bounds.

Recently, Generative Methods that learn easy to sample representations of dataset's hidden probabilistic distribution have gained attention due to successes in fabricating synthetic data, in particular data of faces of humans that have never walked this world [3]. As pointed out by Morningstar and Melko in [4] Generative Models such as Variational Autoencoders an interesting alternative to the canonical MCMC based approaches. This is because Variational Autoencoders on top of achieving our goal of being able to sample without correlated variable

will also provide a probabilistic model of the data and a compact representation of the latent space [5]. In this project we will test out this idea. We will explore if we can use a semi-supervised learning technique (Variational Autoencoders) to learn the distribution of a Condensed Matter Physics model: The 2D Ising Model for ferromagnets.

Methodological Approach

The Ising Model is one of the few non-trivial models that due to the mathematical prowess of Lars Onsager [6] has a nice analytical expression for the expectation values of the observables. Our goal for this project will be to learn the magnetization of the Ising Model with Variational Autoencoders. To accomplish this, we will based our approach to the one found in [7]. First, we will use the Metropolis-Hasting Algorithm to generate pictures of the Ising Model as well as a computational benchmark for expected magnetizations at different temperatures. We will use the results from Metropolis-Hasting as benchmark and Onsager's curve only as reference because Onsager's solution, even though approximates all sizes decently well below a critical temperature, is exact only for the case were the dimensionality of the Ising system approaches infinity [6].

We will implement the Metropolis-Hasting algorithm and generate a data set of 5000 28x28 Ising Lattice pictures in the range of temperatures from 0.01 to 5K in steps of 0.01K with 100 pictures per temperature and 100 warms up iterations per picture. We will average out the magnetization per spin over the 100 pictures of the same temperature and report that value obtained for magnetization at all temperatures as our computational benchmark.

Once we got the data and the benchmark, we will proceed to use this data to train two different architectures of Conditional Variational Autoencoders (a twist on classic VAE's that learn distribution conditioned on an inputted label that will be the temperature). These two architectures will be a "vanilla" fully connected dense C-VAE and a convolutional C-VAE (a Conv-CVAE).

Our hyperparameter will be the dimensionality of the latent space and starting at a dimensionality of two we will explore different values and validate against the Metropolis-Hasting values for magnetization. This will be done by sampling 100 pictures for each temperature for each value of the dimensionality and taking the magnetization for each, averaging out over the batch of 100 Ising pictures with the same temperature.

Theoretical Framework

The Ising Model

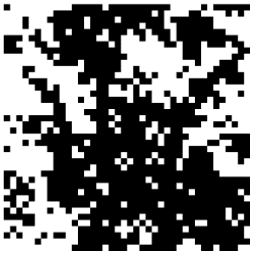


Fig 1: An example of a 2D Ising Picture each black pixel represents a lattice variable in the +1 state and each black pixel represents a lattice variable in the -1 state

The Ising Model abstracts the physics from magnets by representing the alignment of magnetic dipoles of atoms, neutrons, electrons, etc, as an N-site square lattice with N*N binary variables $\sigma \in \{+1, -1\}$ [2]. An Ising system is usually represented using pictures where white pixels represent a variable taking the value of +1 and black pixels represent a variable taking the -1 value (see figure 1). The total energy of the system in a given configuration of the variable also known as the Hamiltonian is given by the nearest neighbor interaction of all dipoles where coaligned neighbors favor lower energy configurations. Mathematically this is represented by:

$$H(x) = -\sum_{\langle i,j \rangle} \sigma_i \sigma_j$$

Where the X variable represent the configuration of the entire system all the binary variables in

their respective lattice positions, so the sum is over all $\sigma's$. The energy of the system is relevant because it dictates the probability of finding the system in that given configuration. This mapping from configurations to probability is given by the Boltzmann distribution which is the following:

$$P(x) = \frac{\exp(-H(x)t^{-1})}{\sum_x \exp(-H(x)t^{-1})}$$

The magnetization at any given configuration is calculated as:

$$M(x \mid t) = \sum_i \sigma_i$$

And correspondingly the expected magnetization can be expressed as:

$$\langle M(t) \rangle = \sum_x M(x \mid t) P(x \mid t)$$

Finally, the formula found by Onsager when the dimensionality approaches infinity is

$$\langle M(t)_{onsanger} \rangle = (1 - [\sinh(2t^{-1}) \sinh 2(2t^{-1})]^{-2})^{1/8}$$

Variational Autoencoders

Variational Autoencoders are a type of Generative model that seeks to recreate the distribution of the input, P(X), by assuming that there exist a set of overarching hidden variables (the "z" latent variables) that through a deterministic mapping parametrize by some vector theta which is often through a neural network can approximate the distribution of X [8]. It is assume by the algorithm that Z has a Normal form

VAE's exploit the fact that the probability of recreating x from z : $p(x \mid z; \Theta)$ will be close to 0 for most z's. Thus, it takes a shortcut by first learning a distribution $Q(z \mid x; \phi)$ that by tuning the parameters phi of a neural network (often called the "encoder" network) learns the distribution of the values of z that are likely to happen given an input x. The idea is to use this reduced distribution to sample z.

The other term $p(x \mid z; \Theta)$ is also implemented as a neural network and it is often called the "decoder". It propagates gradient comparing the outputted x with the original x inputted to Q [**2**] using some metric of difference that in our case will be the pixel-wise cross-entropy loss.

The next step for VAE's is to exploit the fact that $\mathrm{E}_{z \sim Q}[p(x \mid z; \Theta)]$ and P(X) are related through:

$$\log(p(x)) - D[Q(z \mid x; \phi) \,||\, P(Z \mid X)]$$
$$= \mathrm{E}_{z \sim Q}[\log(p(x \mid z; \Theta))] - D[Q(z \mid x; \phi) \,||\, p(z)]$$

Where D denotes the KL Divergence. We can make the second term on the right approach 0 by choosing an arbitrary high-capacity Q [8] remaining only:

$$\log(p(x)) = \mathrm{E}_{z \sim Q}[\log(p(x \mid z; \Theta))] - D[Q(z \mid x; \phi) \,||\, p(z)]$$

Now using our assumption of z having a gaussian form and reparametrizing the sampling from Q so we can propagate gradient through the stochastic process we can finally get our loss function:

$$L(\Theta, \phi) = \mathrm{E}_{e \sim N(0,I)}[\log(p_\Theta(x \mid \mu(x) + \Sigma^{1/2}(x) * e; \Theta))]$$
$$-D[Q_\phi(z \mid x; \phi) \,||\, N(0,1)]$$

Inspecting our loss function the first term can be interpret as a reconstruction loss and the second term as build in regularization that pulls our Q to be as close as a normal as possible. We will follow the most conventional chose of Q such that**:**

$$Q(z \mid x; \phi) = N(z \mid \mu(x; \phi), \Sigma(x; \phi))$$

This will allow us to have a nice, closed form expression for the KL-divergence [8].

Finally, because we want to learn a different reconstruction that depend on temperature, we want to manually add a parameter to our decoder network as follows:

$$L(\Theta, \phi) = \mathrm{E}_{e \sim N(0,I)}[p(x \mid \mu_\Theta(x) + \Sigma_\Theta^{1/2}(x) * e, t)]$$
$$-D[Q(z \mid x; \phi) \,||\, N(0,1)]$$

The reader can better understand this process by referring to **Fig2** which contain an illustration extracted from towardsdatascience.com in a post authored by Md Ashiqur Rahman [8]

Fig 2: Example graph of Conditional Variational Autoencoder

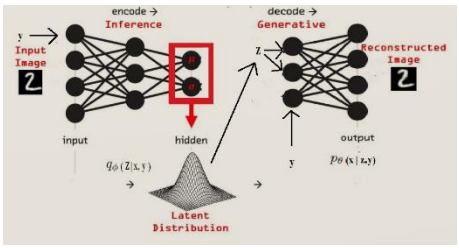


Fig2: Shows a high-level diagram of CVAE extracted from https://towardsdatascience.com/understanding-conditional-variational-autoencoders-cd62b4f57bf8

Metropolis-Hasting Algorithm

As pointed out by Koetz in [2] Metropolis et al change conventional thinking in the 40s by suggesting that complex distributions could be sampled by using Markov Chains. Under some conditions Markov Chain converge to a given distribution can be crafted [9] and thus the Metropolis-Hasting algorithm uses this fact a creates a chain that fulfill this convergence conditions. The algorithm starts the Markov Chain at a random state and according to the difference of the probability densities with a candidate new state, deterministically accepts and move to the new candidate state if it has higher probability density than the current state or in some cases even if it has lower density it accepts and move to the candidate state based on some likelihood proportional to their difference.

The following flow chart also found in [2] explains the algorithm applied to the Ising Model in detail. The reader should interpret difference in energy in the chart as differences in the Hamiltonian and should set the constant k=1 for our case.

Fig 3: Flowchart of the Metropolis-Hasting algorithm

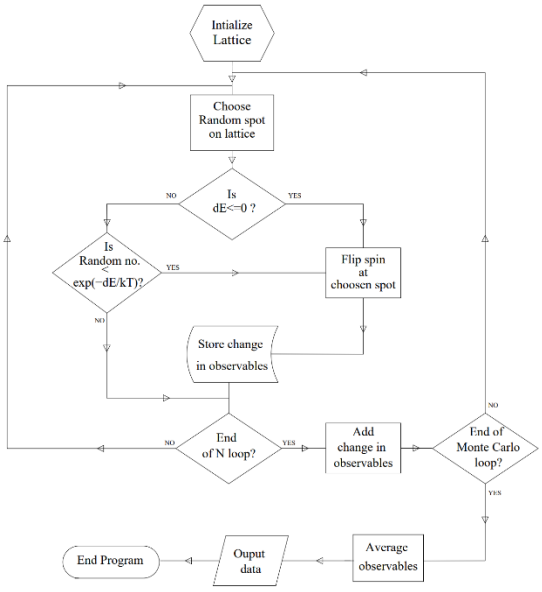


Fig3: Shows the flowchart found in [2] authored by where the Metropolis-Hasting algorithm is explained. In our case we set k=1

Experimental Results

Benchmark

The computed Metropolis-Hasting algorithm results and the analytical Onsager's curve we got are shown below

Fig 4: Magnetization values for expected magnetization per spin

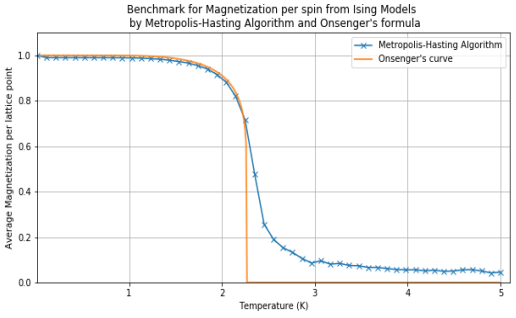


Fig4: shows the results of Metropolis-Hasting algorithm for the expected magnetization of the input data and the Onsager's curve

Dense-CVAE

Fig 5: Architecture of Dense-CVAE

Model: "encoder"

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 784)]	0	
dense_10 (Dense)	(None, 32)	25120	input_5[0][0]
dense_11 (Dense)	(None, 16)	528	dense_10[0][0]
z_mean (Dense)	(None, 2)	34	dense_11[0][0]
z_log_var (Dense)	(None, 2)	34	dense_11[0][0]
sampling_2 (Sampling)	(None, 2)	0	z_mean[0][0] z_log_var[0][0]

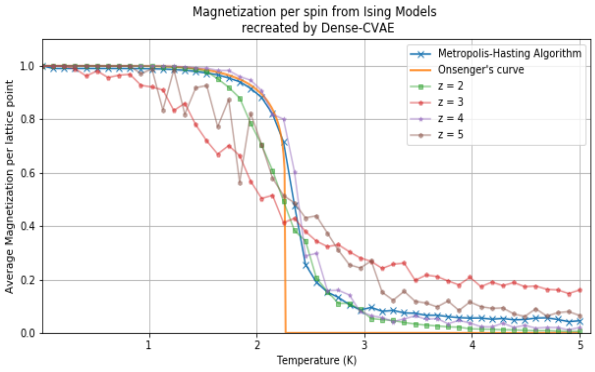
Model: "decoder"

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 3)]	0
dense_7 (Dense)	(None, 16)	64
dense_8 (Dense)	(None, 32)	544
dense_9 (Dense)	(None, 784)	25872

Fig(x) shows a screenshot from the architectures of the encoder and decoder of the Dense C-VAE with z=2. The output of the "encoder" is the input of the "decoder" the extra dimension comes from the temperature that we are concatenating

The architecture for the decoder and encoder network for a our dense CVAE for a latent space of 2 is shown above. The curves of magnetization after training are the following:

Fig8 Magnetization per Spin of generated data with the Dense-CVAE



Conv-CVAE

Fig 7: Architecture of Conv-CVAE

Model: "encoder"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 28, 28, 1)]	0	
conv2d (Conv2D)	(None, 14, 14, 32)	320	input_1[0][0]
conv2d_1 (Conv2D)	(None, 7, 7, 64)	18496	conv2d[0][0]
flatten (Flatten)	(None, 3136)	0	conv2d_1[0][0]
dense (Dense)	(None, 16)	50192	flatten[0][0]
z_mean (Dense)	(None, 2)	34	dense[0][0]
z_log_var (Dense)	(None, 2)	34	dense[0][0]
sampling (Sampling)	(None, 2)	0	z_mean[0][0] z_log_var[0][0]

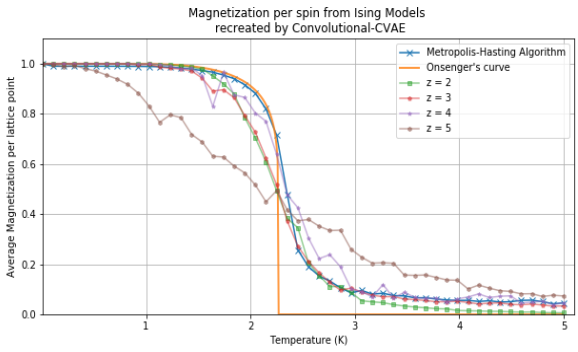
Model: "decoder"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 3)]	0
dense_1 (Dense)	(None, 3136)	12544
reshape (Reshape)	(None, 7, 7, 64)	0
conv2d_transpose (Conv2DTran	(None, 14, 14, 64)	36928
conv2d_transpose_1 (Conv2DTr	(None, 28, 28, 32)	18464
conv2d_transpose_2 (Conv2DTr	(None, 28, 28, 1)	289

Fig(x) shows a screenshot from the architectures of the encoder and decoder of the Convolutional C-VAE with z=2. The output of the "encoder" is the input of the "decoder" the extra dimension comes from the temperature that we are concatenating

Similarly, the architecture for the decoder and encoder network for our Convolutional CVAE for a latent space of 2 is shown in above. The curves of magnetization after training are the following

Fig9 Magnetization per Spin of generated data with the conv-CVAE



Taking the l1 norm between the curve datapoints we get that the best performing hyperparameters for the dense and convolutional architectures are the ones with Z=4 and Z=3 respectively (a 4D and 3D dimensionality of the latent space) achieving errors of 0.000304 and 0.001785 compared to the Metropolis-Hasting generated data. Surprisingly, the Dense architecture outperformed the convolutional architecture.

Fig10: Magnetization data for data generated by best hypermeters per architecture

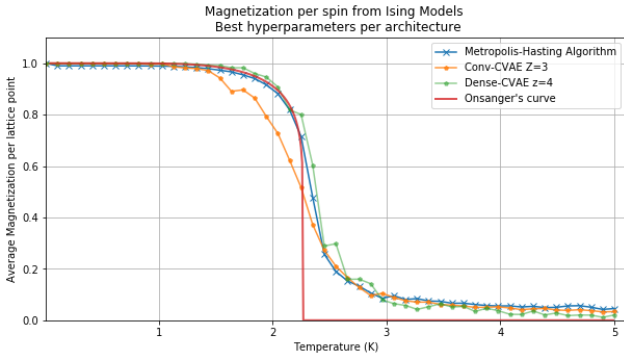


Fig10 the average magnetization of the generated Ising images with the best hyperparameters for each CVAE architecture. The Metropolis-Hasting generated data and the Onsager's curve are also shown

Conclusions

We attempted to learn the magnetization of the Ising model for magnets by using a semi-supervised technique, C-VAEs. We executed the Metropolis-Hasting algorithm to generate Ising pictures and computational benchmark to compare against with. In addition to this we also use the Onsager's curve below critical temperature as a sanity check. We tuned the dimensionality of the latent space as our hyperparameter. The best performing architecture was the Dense fully connected C-VAE with a z dimensionality of 4 recreating the magnetization curve of the computational benchmark with an overall error of 0.0003

References

[1] Naudts J. The q-exponential family in statistical physics. Open Physics. 2009 Sep 1;7(3):405-13.

[2] Kotze, J. (2008). Introduction to Monte Carlo methods for an Ising Model of a Ferromagnet. arXiv: Statistical Mechanics.

[3]Curtó, J.D., Zarza, H.C. and Kim, T., 2017. High-resolution deep convolutional generative adversarial networks. arXiv preprint arXiv:1711.06491.

[4] Alan M and Roger G. M. (2017). Deep learning the Ising model near criticality. J. Mach. Learn. Res. 18, 1 (January 2017), 5975–5991.

[5] Kingma, D. P. & Welling, M. (2014). Auto-Encoding Variational Bayes. 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings,

[6] Onsager (1944), L. Crystal statistics. I. A two-dimensional model with an order-disorder transition." Physical Review 65: 117-149.

[7] Nosarzewski, B., 2017. Variational Autoencoders for Classical Spin Models. http://cs231n.stanford.edu/reports/2017/pdfs/538.pdf

[8]Rahman,M.A Understanding Conditional Variational Autoencoders. (2019). Retrieved from: https://towardsdatascience.com/understanding-conditional-variational-autoencoders-cd62b4f57bf8

[9] Metropolis, Nicholas, Rosenbluth, Arianna W., Rosenbluth, Marshall N., Teller, Augusta H. and Teller, Edward (1953): "Equation of State Calculations by Fast Computing Machines." The Journal of Chemical Physics 21 , no. 6: 1087-1092.